REMARKS

Please reconsider the application in view of the following remarks. Applicant thanks the Examiner for carefully considering this application.

Disposition of Claims

Claims 1-4 and 6-9 are pending in the present application. Claims 1 and 6 are independent. The remaining claims depend, either directly or indirectly, from claims 1 and 6.

Claim Amendments

Independent claims 1 and 6 have been amended to clarify that the access control level is associated with the declaration of the object member, and to correct typographical errors. Applicant respectfully asserts no new matter has been introduced by these amendments, as support may be found, for example, in paragraphs [0097-0101] of the specification.

Attorney Docket Number

Applicant respectfully requests that the Attorney Docket No. for this matter be changed from "0007056-0197/P5940" as indicated on the cover sheet received with this Office Action to "16159/092001; P5940."

Rejection under 35 U.S.C. §103

Claims 1-3, 6-8 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,579,518 (hereinafter "Yasumatsu") in view of U.S. Publication No. 2005/0172302 (hereinafter "Chan"). To the extent that this rejection applies to the amended claims, the rejection is respectfully traversed.

The claims of the present invention are directed to a run-time determination of access rights to an object member in a dynamically typed programming language. In this context, being dynamically typed means that class members, methods, and virtual methods gain the benefit of late binding as described in the specification. For example, a class member may be declared without a type prior to compilation, the type being subsequently determined at run-time depending on an assigned value (*see* Specification, paragraphs [0071-0081]). Further, the determination of access rights depends in part on an access control level associated with the declaration of the object member. Specifically, the object member may be declared as public or private, with access rights then depending, at run-time, on the declaration and the calling block's relationship to the object member. For example, if the object member is declared as public, then it may be accessible to all blocks. Alternatively, if the object member is declared as private, then it may be accessible only to its own block. Those skilled in the art will appreciate that public and private are not the only possible access types, as an object member may also be declared as protected, for example (*see* Specification, paragraphs [0097-0102]).

Turning to the rejection of the claims, to establish a *prima facie* case of obviousness "... the prior art reference (or references when combined) must teach or suggest all the claim limitations" and "all words in a claim must be considered in judging the patentability of that claim against the prior art" (*see* MPEP §2143.03). Applicant respectfully asserts that the references, when combined, fail to teach or suggest all the limitations of amended claim 1.

Amended independent claim 1 recites, in part, "declaring said object member in a program written in a dynamically typed programming language, said declaring comprising assigning an access control level to said object member wherein a public member and a private member have different access rights" and "determining, during run-time, whether said object member is accessible using the access control level." The Examiner admits that Yasumatsu does

not teach or suggest all these limitations (*see* Office Action dated January 26, 2006, page 3). Instead, the Examiner has attempted to use Chan to supply that which Yasumatsu lacks.

Specifically, Chan is directed to using existing class functionality in a prior art object-oriented programming language, such as Java™ or C++, to control access to a sensitive function in a class (*see* Chan, paragraph [0022]). That is, access control is implemented in the prior art programming language using "friend object" classes, which are responsible for determining access rights to the sensitive function. The names of packages, classes, and functions may be registered with a friend object via a traditional method call to grant access to the sensitive function. Subsequently, a checkFriend method in the friend object is called as the first statement of the sensitive function to determine the access rights of the calling statement (*see* Chan, paragraph [0035]). Thus, access rights may change dynamically based on *registration entries* provided to the friend object at run-time. Conversely, in the present invention, object members may be *declared* as public or private, with access rights then depending on the declaration and the calling block's relationship to the object member.

For example, as discussed above, an object member declared as public may be accessible to all blocks, while an object member declared as private may only be accessible to its own block. Clearly, run-time *registration* with a friend object is substantially different from an object member *declaration*. Thus, in Chan, there is no teaching or suggestion of the "public member" and "private member" of the present invention. Effectively, to equate the access control of Chan with the public and private members of the present application, the Examiner would be required to read out an express limitation of the claims.

Further, amended claim 1 recites, in part, "binding, at run-time, said object member to its reference if said object member is used and if said object member is accessible." As discussed above, Chan teaches calling a checkFriend method in the friend object as the first statement of

the sensitive function to determine the access rights of the calling statement. Clearly, for the first statement of the sensitive function to be executed, the sensitive function must already be bound to its reference. Thus, the reference binding in Chan occurs *prior* to the access rights determination. However, the claims of the present invention are directed to binding the object member to its reference "*if said object member is accessible*" (emphasis added). In other words, the reference binding in the present invention only occurs *after* the access rights determination. Clearly, the reference binding cannot occur both before and after the access determination. Thus, in order to equate the access control of Chan with the public and private members of the present invention, the Examiner would be required to read out an express limitation of the claims.

In view of the above, Yasumatsu and Chan, whether viewed separately or in combination, do not teach or suggest all the limitations of claim 1. Accordingly, claim 1 is patentable over Yasumatsu and Chan for at least the reasons stated above. The Examiner asserts that claim 6 contains substantially the same limitations as claim 1 (*see* Office Action dated January 26, 2006, page 4). Thus, claim 6 is allowable for at least the same reasons. Claims 2-3 and 7-8 depend, either directly or indirectly, from claims 1 and 6 and are allowable for at least the same reasons. Accordingly, withdrawal of this rejection is respectfully requested.

Further, to establish a *prima facie* case of obviousness, some suggestion or motivation must exist, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings (*see* MPEP, §2143). Applicant respectfully asserts that there is no motivation to combine the teachings of Yasumatsu with the teachings of Chan.

Specifically, as discussed above, Chan is directed to using existing class functionality in a prior art object-oriented programming language, such as Java™ or C++, to control access to a

sensitive function in a class. Those skilled in the art will appreciate that in the prior art programming languages relied upon by Chan, only virtual methods have the benefit of late binding – all other class members and functions are bound at compilation. *See* Chan, paragraph [0022], wherein Chan discloses Java™ and C++ as preferred embodiments, and paragraph [0036] of the present specification, providing detailed discussion of the lack of late binding in these languages, except for virtual methods.

However, Yasumatsu is directed to an object-oriented language wherein types are determined dynamically (*see* Yasumatsu, col. 2, lines 48-50). In other words, Yasumatsu is directed to a programming language providing late binding (*i.e.,* after compilation, during run-time) of object types. Thus, Yasumatsu cannot possibly make use of the prior art programming languages relied upon by Chan, since those languages only teach late binding of virtual methods. Thus, Yasumatsu, effectively, teaches away from the use of the programming languages relied upon by Chan.

In view of the above, regardless of whether the teachings of Yasumatsu and Chan *can* be combined, there is clearly no suggestion or motivation set forth in either Yasumatsu or Chan to do so. Absent such a suggestion or motivation, the teachings of Yasumatsu and Chan cannot be conveniently combined to render the claimed invention obvious. Accordingly, withdrawal of this rejection is respectfully requested.

Further, to establish a *prima facie* case of obviousness, a reference is considered analogous only if it is "in the field of applicant's endeavor or ... reasonably pertinent to the particular problem with which the inventor was concerned" (*see* MPEP, §2141.01(a)). Applicant respectfully asserts Chan is non-analogous art and is improperly used in this rejection.

As discussed above, Chan is directed to using existing class functionality in a prior art object-oriented programming language, such as Java™ or C++, to control access to a sensitive

function in a class (*see* Chan, paragraph [0022]). Those skilled in the art will appreciate that friend objects as described above simply make use of existing class functionality without adding any new features to the programming language itself. Conversely, the present invention is directed to *integrating* access control features into a dynamically typed programming language. Specifically, in the present invention, declaring an object member public or private is functionality associated with keywords in the programming language itself, *not* simply a data structure using existing class functionality to extend an existing programming language. Clearly, using *existing* class functionality of a prior art programming language, such as Java™ or C++, to extend access control is not analogous to *integrating* access control features into a dynamically typed programming language.

In view of the above, Chan is non-analogous art and cannot be legally be used as asserted in this rejection. Accordingly, withdrawal of this rejection is respectfully requested.

Claims 4 and 9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yasumatsu and Chan, in view of Admitted Prior Art (hereinafter "APA"). To the extent that this rejection applies to the amended claims, the rejection is respectfully traversed.

As discussed above, Yasumatsu and Chan, whether viewed separately or in combination, do not teach every limitation of independent claims 1 and 6. Further, APA fails to provide that which Yasumatsu and Chan lack, as evidenced by the fact that the Examiner has relied on APA solely to teach "wherein said class method is a virtual method" (*see* Office Action dated January 26, 2006, page 5). Thus, independent claims 1 and 6 are patentable over Yasumatsu, Chan, and APA for at least the reasons stated above. Claims 4 and 9 depend, either directly or indirectly, from claims 1 and 6, and are patentable for at least the same reasons. Accordingly, withdrawal of this rejection is respectfully requested.

Conclusion

Applicant believes this reply is fully responsive to all outstanding issues and places this application in condition for allowance. If this belief is incorrect, or other issues arise, the Examiner is encouraged to contact the undersigned or his associates at the telephone number listed below. Please apply any charges not covered, or any credits, to Deposit Account 50-0591 (Reference Number 16159/092001).

Dated: April 26, 2006

Respectfully submitted,

By OSHA #L0031
for Robert P. Lord
Registration No.: 46,479
OSHA · LIANG LLP
1221 McKinney St., Suite 2800
Houston, Texas 77010
(713) 228-8600
(713) 228-8778 (Fax)
Attorney for Applicant